ISRN SICS-T--94/05-SE

Adaptive Help by Navigation and Explanation

by

Benoit Lemaire, Catriona MacDermid and Annika Wærn

SICS Technical Report
T94:05
ISSN 1100-3154

Report T94:05

ISRN: SICS-T--T94/05--SE
ISSN : 1100-3154

ADAPTIVE HELP BY NAVIGATION AND EXPLANATION

BY

Benoit Lemaire, Catriona MacDermid and Annika Wærn

UACM; 940419

{lemaire, catriona, annika}@sics.se

Swedish Institute of Computer Science
Box 1263, S-164 28 KISTA, SWEDEN

ABSTRACT

Adaptivity in help systems can be achieved either by intelligent help techniques, but also by very flexible and easily controlled user interface mechanisms. In this paper, we discuss an outline of an interface that combines these two approaches to adaptivity. The fundamental property which enables this integration is that the necessary knowledge is both explicitly represented for explanation generation, and {\it explicitly presented to the user} for navigation and manipulation.

The success of intelligent help requires that a highly user-oriented approach to the design of the interface is used. The interface described here is being developed for a help system which is intended to aid program developers in their use of a development methodology, which is an unusually advanced target for adaptive help. We describe our strategy for capturing usability requirements as a combination of three methods: investigating the users' help needs in the current context, identifying general properties of computer interfaces that apply to the identified problems, and finally investigating the usability properties of a selected design solution through rapid prototyping.

The paper reports the findings from the initial interviews, and describes the selected basic solution as realised in the prototype. We also describe how this basic structure can be enhanced in different ways, in order to yield different adaptation mechanisms.

KEYWORDS: Adaptive interfaces, intelligent help, explanation, user oriented design

Adaptative help by navigation and explanation

Benoît Lemaire Catriona MacDermid Annika Wærn

Swedish Institute of Computer Science
Box 1263
164 28 Kista

e-mail: {lemaire,catriona,annika}@sics.se

1 Introduction

Adaptive help has several advantages over traditional help systems: it can give help even when the user has difficulties describing exactly what help he or she needs; it can select appropriate information in a large and complex information domain; and it can tailor the presentation of information to the particular user. However, adaptiveness can be reached through different means, both by intelligent help techniques, but also by very flexible and easily controlled user interface mechanisms. In this paper, we discuss an outline of an interface that combines these two approaches to adaptiveness.

The help interface is being developed as part of a project aiming to create a help system for designers working with an extremely advanced development methodology (NDM), geared to the development of large scale, high-performance and modular systems. In this project, we are using a user-oriented approach to system development. The user-oriented approach to system development is extremely important for the development of user-adaptive help, as you otherwise run the risk of allowing for the wrong adaptations, and missing out on adaptation mechanisms that are crucial.

The system development was initiated from an extensive survey of experienced difficulties with the methodology NDM. Based on these studies, we have identified some essential requirements for the target help system. These have led us to define and implement a first prototype system, in which the essential requirements have been met, or can be supported by extensions of the prototype. The design is based on a combination of flexible user access to information, and intelligent help mechanisms for context-dependent answer generation, clarification sub-dialogues etc. This prototype will serve as a basis for the identification of further requirements, such as requirements on clarifications and identification of missing information types.

This paper reports the findings from the initial interviews, and describes the selected basic solution as realised in the prototype. We also describe how this basic

structure can be enhanced in different ways, in order to yield different adaptation mechanisms.

2 Getting requirements from user interviews

The first stage of the project has been the initial identification of help needs among users of the new software development methodology, NDM. This was achieved through in-depth and focussed interviews with users. Most interviews were conducted with software designers in three design projects currently using NDM. Interviewees ranged from those who had only recently had initial training in NDM to those who had had practical experience of an entire design project using the process and those who had several years' experience of design projects using the previous software development methodology.

Fifteen people were interviewed. Interviews lasted between thirty minutes and two hours and most were tape-recorded. Most interviews were conducted with individuals, but two interviews were conducted in small groups. Principally these interviews yielded data in terms of the problems encountered with NDM in learning how to use it and applying it in practice; in addition they revealed the problems encountered with the currently available help sources (which provides guidelines for features of an alternative help system). The main findings are outlined below.

2.1 Problems in applying the NDM methodology

2.1.1 User-orientation

A major criticism that many users had was that they felt that NDM was too abstract. This related to a difference between the theoretical process and what happens in practice. The problem was particularly acute for designers who were familiar with the (more procedural) methodology used previously by the organization. These users clearly need support at first to appreciate the different way of working.

Typical questions users had were of the following types :

- Shortcuts e.g. *How much do I need to read ?*
- Rationale e.g. *Why do I have to produce this document ?*
- Procedures e.g. *How do I produce this document ?*
- Navigation e.g. *Where should I put this document ?*
- Completion criteria e.g. *When should I stop the IOM stage ?*

2.1.2 Examples

A very common request was that greater priority should be given to the provision of examples to illustrate concepts and to draw distinctions between different parts

of the process. It was claimed that "examples could solve all our problems". Existing examples were hard to interpret, since it was not made explicit whether they were theoretical or real-life examples. Simple, theoretical examples should introduce each concept, followed by several more complex examples selected from real project documentation.

One critical decision causing users problems was knowing when a modelling step is completed. Guidance is required on this, using realistic examples to illustrate how to make the judgement.

New terminology caused users confusion. This needs to be clearly defined and illustrated with multiple examples.

2.1.3 Omission criteria

Knowing what to document is a major concern for project members. Further clarification about what documentation is compulsory and what is optional is needed. Designers state that they need to know the purpose of the documentation when producing it.

Users are somewhat inclined to omit parts of the process in cases of uncertainty. The desire to omit modelling stages is seen to be a result of the similarities between some of the modelling stages (such as the IOM and ROM processes) and a reluctance to produce too many overlapping documents. Clear guidelines on taking shortcuts in the process should be made available, but this is likely to depend on the project context.

2.2 Problems with existing help sources

Both the hard-copy manuals and the existing online help system for NDM were criticised because they were cluttered with standardised definitions. Interviewees felt that the amount of documentation was overwhelming and they needed help which was specifically geared to their needs. The online help was the most popular, particularly with novice designers who had had an opportunity to explore it during the course flow. Others had to learn how to use it by trial and error, tended not to be able to navigate around the whole help and may thus have under-estimated its possibilities.

It appears that designers have quite basic questions about the methodology which they are too embarrassed to air in public. It would appear that they are not able to find these answers in the online help. Where help of this type is not available, it will be produced at grassroots level in the form of cookbooks, but this is time-consuming for designers and may not always provide the best possible help, since it may deviate from the NDM documentation.

One designer on each project is assigned to compile and submit trouble report forms to the NDM developers, but because trouble reports get delayed feedback, interviewees tend to ask a colleague for help instead rather than consult official documentation. All the interviewees relied heavily upon informal discussions. These were actually favoured over all the other sources of help. The implication of

this is that sometimes the problem doesn't get documented and only an unofficial, ad-hoc solution exists, becoming part of the designers' 'folklore'.

3 Getting requirements from computer capabilities

In summary, the user interviews emphasized the following points:

1. Problems in applying the NDM methodology:
    - designers often ask questions about shortcuts, rationale, procedures, navigation and completion criteria. They find the methodology too abstract and theoretical for their purposes and request explanations and real-life examples of how it works in practice.
2. Problems with existing help sources:
    - there is a need for navigating through the current help information in a better way;
    - the current help is not contextual.

Based on experiences we have as well as references in the human-computer interaction literature, we can reinterpret the previous needs within the context of a computer interface. We will see that some requirements come directly from the user requirements analysis whereas others arise from known features of human-computer interaction and computer functionality that has no equivalent in the human world.

3.1 Navigation versus explanation

The first new requirement that we will put on the help system comes from the need of navigation support. Help is often viewed as a piece of information that is provided by a system to a user. But allowing a user to navigate through the system knowledge is also a form of help. We thus distinguish between two different forms of help: user navigation versus system explanation. In the former, the user is autonomous and looks himself or herself for help by navigating through the information displayed whereas in the latter the user queries the system which provides him/her with a help solution. Both forms complement each other; let us discuss each one in detail.

Navigation Allowing the user to navigate through the system knowledge is giving him/her access to this knowledge directly, without having to query the system. This way of getting the right information is obviously more efficient when the user knows how to do it.

Usually, the user does not know how to get the information because this information is not accessible to him/her. In such a case, the only way of getting help is to ask a question. However, if the knowledge is represented on the screen

in such a way that the user can browse it, he or she does not have to query the system, but just looks at the information available, and therefore has full control.

Another advantage of representing the knowledge explicitly on the screen is that questions can be composed easily, by pointing to objects on the screen. For instance, instead of typing

```
compare rom_process and iom_process
```

the user can type or select in a menu the item *compare*, then point to the two processes if they are accessible on the screen. This kind of interaction has been used in particular to facilitate user's follow-up questions [Moore and Swartout 90].

In a study comparing direct manipulation to natural language, Cohen argues that one of the problems with direct manipulation techniques is that they can only be used for information that is represented on the screen [Cohen 92]. Indeed, not everything can be accessible to the user. In such cases, explanation is necessary.

Explanation As we said before, users also expressed the need to ask questions about the methodology. In fact, navigation cannot solve all of the user's problems even though the range of questions is diminished because of this new form of help. Some users' needs require contextual help that only explanation can provide. When users cannot find the information they want, either because it cannot be represented on the screen or because they do not know where it is, they query the system. This system will access the domain knowledge to compose an answer tailored to the context. Although the information represented on the screen can also be contextualized, explanation provides help which is much more tailored to the context than that which comes from pure navigation. The reason for this is that most of the navigation information is constructed beforehand whereas explanation is built according to the current context.

Forms of knowledge representations It is worth noting that because of the dual form of help (navigation versus explanation), knowledge has at least two different forms of representations :

- a visual representation that the user can navigate;
- a formal representation that the explanation generation mechanisms use.

It is widely recognized that explanation requires explicit knowledge representations for the system purpose. In the same way, navigation requires an explicit representation of the knowledge *on the screen*, for the user purpose. Navigation and explanation both require representations.

3.2 Visual versus textual information

The other requirement we decided to put on the system comes from the real domain as well as from the computer point of view. The current methodology documentation relies heavily on graphics and this kind of information has

	textual information	graphical information
user's navigation	navigating in previous answers	navigating in graphs
user's query	typing a request	clicking on graph nodes, words, pieces of texts
system's explanation	providing texts	highlighting text, highlighting graph nodes

Figure 1: Different activities involved in the human-computer interaction

to be represented in the help system. Another justification of this requirement comes from the computer point of view: because our goal is to design a computerized help system, we will be able to rely on the particularities of the technology and present the information on different media. This view is based on experiences and literature that comes from the resources available. Many authors have pointed out the richness of multimedia explanations [Bretan and Karlgren 93, Feiner and McKeown 91, IMI 93, Wahlster et al. 91]. We decided to rely on textual and graphical information. The number of potential explanations is doubled that way. For instance, a comparison between *session diagram* and *interaction diagram* can be either a text stating the differences and similarities between the two entities, or two views presenting the properties of each object with the common parts highlighted. In the same way, a description of the IOM process can be a textual description as well as a picture of the processes graph showing where IOM appears in the hierarchy. Querying the system can be typing a text or selecting visual objects with the mouse and choosing a question item in a menu. Having multiple strategies to answer the same question has been recognized as an important feature of explanation systems [Lemaire and Safar 91].

In our domain, the visual information we have to display is mainly graphs: objects involved in the methodology and their relations. The textual information is answers to users' questions but also the dialogue history, which contains all of the previous interaction between the user and the system. Our previous experience showed us the need for representing on the screen the previous information given to the user [Lemaire and Moore 94].

According to the requirements we have established so far (navigation versus explanation and textual versus graphical help), we can describe the different activities involved in the interaction between the human and the computer, as shown in figure 1.

3.3 Context-dependent help

Several of the demands expressed by the users can be addressed only if the full context of the user's question is taken into account. The following issues expressed by users fall in this category.

- Examples. A very common request was that more examples should be given, and that these should be "less abstract", "taken from real projects" etc. Examples are only useful when their content by analogy applies to the user's own problem, and the voiced discontent with the existing examples shows that this seldom happens by itself. It is fairly easy to collect examples, by collecting previous results of applying the methodology, but less obvious how to select an example that resembles the user's current problem. In particular, this requires that the system can collect and represent the context in which the user demands an example.

- Omission criteria. The definite "yes" or "no" answer to an omission question depends on the characteristics of the user's project. This occurs because omission criteria is a particular case of *guideline* information. In general, guideline information describes what to do and how to do it, given that a particular situation has occurred. If no context is known, a guideline or omission question can only be answered by a vague "yes if ..., but no if ...". If the context is known, this can be used to provide definite answers to help requests, but also to actively produce help and hints.

- Help dialogue. The dialogue with the help system itself can utilize contextual information. The interpretation of a vague help request can be aided by contextual information, and the contextual information can also contain knowledge about previous help requests and relate new answers to old ones.

We expect to include three kinds of contextual information in the help system.

- The status of the project that the user is involved in. This project status constitutes a "global context", and is common to a whole development team.

- The user's current activity and goals. This constitutes a "local", user-specific context, and can be explicitly entered or guessed from the user's work in some other tool, or even from his/her interactions with the help system itself.

- The dialogue history. This constitutes a discourse and linguistic context, that can be used both for explicit references in questions and answers, and to resolve linguistic ambiguities.

In the same way as the system knowledge must be represented explicitly both for the system (to enable explanation) and the user (to enable navigation), so must contextual information be accessible both to the system and to the user. The user must both be able to see and make references to all three types of context.

4 Getting requirements from prototype testing

So far we put requirements on the system from two *a priori* knowledge sources: human studies and computer capabilities. But this is still not enough to get all the specifications of a system designed to interact with a human. Some requirements cannot be imagined beforehand because the context of a human interacting with

the system does not exist yet. And the creation of this context will lead to new requirements: users will react in ways we could not have predicted, having them manipulating the system will make new needs appear, etc. New requirements will come only after the system is created. In order to deal with that particularity, we have to design a rough prototype, put users in front of it, gather their and our new requirements, and start the cycle again. For the reasons we mentioned so far, this rapid prototyping method is widely used in human-computer interface design. It is worth noting that a help system should be based in a modular way, because of this way of development. It should be easy to modify it, add new features, change old ones, in an iterative process.

We have implemented a prototype in Prolog, using `Sicstus Objects` (a mix of logic and objects) to represent the information about the methodology.

4.1 The interface

As we said before, the interface is based on two ideas. The first one is to display information on the screen so that the user can access it directly, and navigate through it. The second idea is to mix graphical information and textual information. Figure 2 shows the interface, with the graphical information in the top and the textual information in the bottom.

Before going into details, let us describe briefly the structure of our domain. There are two main types of entities: processes and objects. Processes are steps in the software design methodology whereas objects are specifications, tables, documents, pieces of software, etc. Processes are composed of sub-processes; they take objects as inputs and they produce new objects as outputs. Relations exist between objects.

Following this structure, our graphical display is composed of two views: a process view and an object view.

The current process shown on the process view is `SubD`. Its subprocesses are displayed on the bottom, its inputs on the left and its outputs on the right. Every node is a button so that the user can click on it. Clicking on a subprocess will go one level down. Clicking on an object will display it on the object view.

On the object view, the current object shown is `sti_subsystem`. One can see the process of which it is the output (`rom`), and the process of which it is the input (`rom`). The five nodes that are linked to `sti_subsystem` are shown with the names of the relations. Clicking on an object node will redisplay the graph from this node. Clicking on a process node will display it on the process view.

These two minimal graph browsers allow the user to navigate both the process structure and the object structure and to go from one to the other.

The textual information is composed of three views devoted to the user's question, the answer and the dialogue history. The question is either typed by the user or a template is selected from a menu. In our example, the user has selected the template `compare(?,?)`. The user can then select objects in any views to complete the question. It means he or she can select a node from one of the two graph browsers, but also select a word from any of the textual views. For instance, selecting the word `iom` in the dialogue history view is exactly the same as selecting

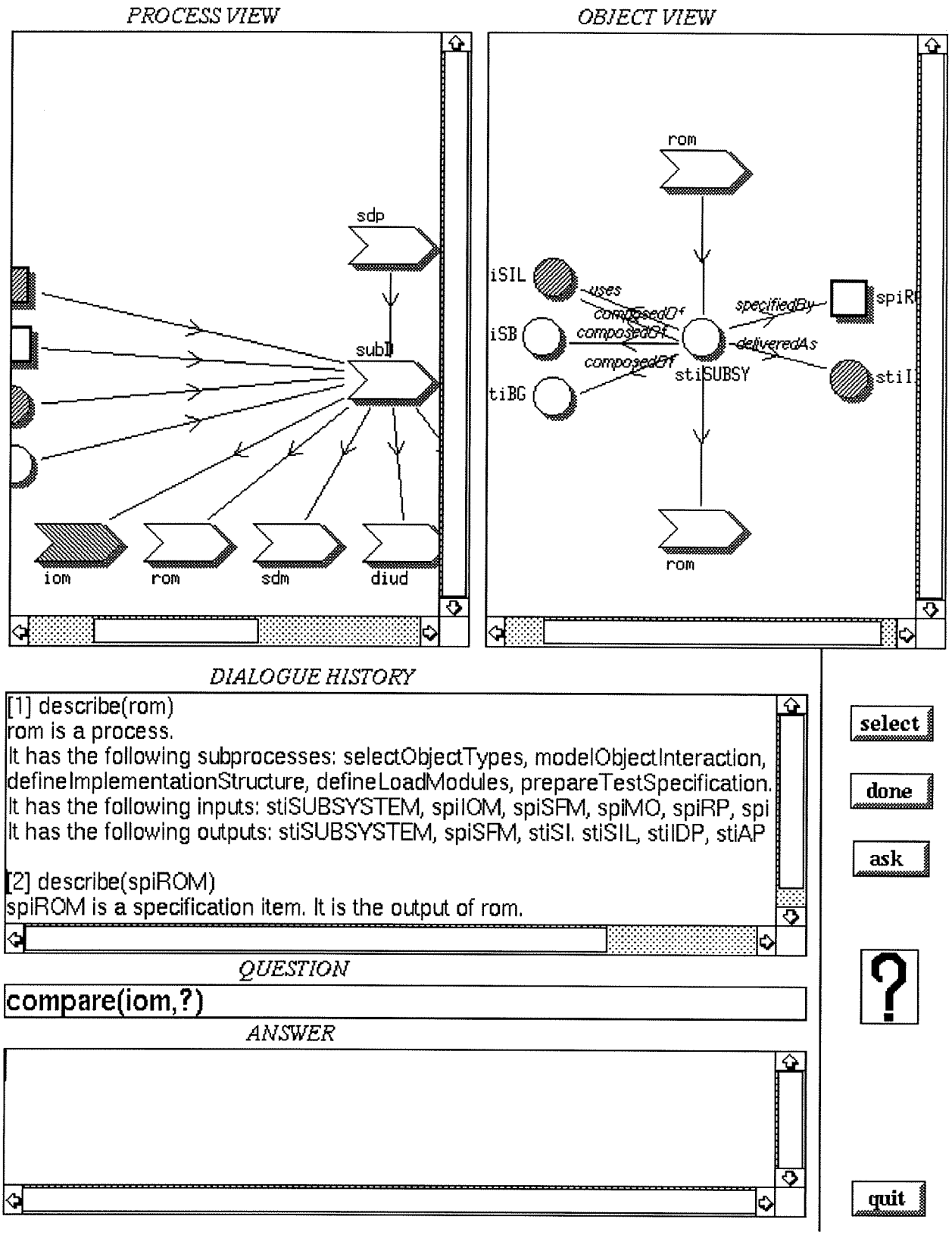

Figure 2: Screendump of our prototype

the node `iom` in a visual view. In our case, the user has selected the process `iom` by clicking on it in the process view.

The dialogue history view displays the previous question/answer pairs.

4.2 Explanation generation

The current explanation is generated by means of operators attached to objects. It does not take into account contextual information yet, but just rewords the visual information. Generating context-sensitive explanations is our next step. We also intend to generate multi-modal explanations (for instance, supplementing a text about a process with a highlight of the corresponding graph node). The idea is to use explanation planning mechanisms and to mix both textual goals and visual goals into the planning operators, similar to [Maybury 91]. Requirements on explanations will be taken from user feedback on the prototypes as well as from the collection of human help dialogues.

4.3 Contextual information

Of the three context types, the current demonstrator addresses only two, the global context and the dialogue history. Both these are represented explicitly for the system, as well as accessible for users. The explicit representation of a global context makes it possible to connect the system to external sources, that can provide information about a project. Currently, the system makes no use of the information for explanation generation, but since it is presented and can be manipulated by the user, it still serves a purpose in the prototype.

- The global context is currently represented as sets of states for each type of process or object. It is presented by colour-marking process steps and objects dependent on their status, and users can change the status of a process or object.

- The dialogue history is currently accomplished in the dialogue history window, but this shows only the history of help requests. Dialogue contextual information should also be present in the graphics interface. We could for example display an additional graphical window showing the current position in the overall knowledge structure, or always present an object in a context, which is dependent on what was previously presented.

We have not yet worked out a representation of a local context. Some information must be added to the knowledge base. Our hope is that information about the intended purposes of singular process steps can be structured to form a goal hierarchy over the individual process steps. Such information can serve as a basis for representing and recognising local contextual information. Since we recognized in section 2 that users express a wish to known *why* particular activities have to be pursued, goal information will this way serve a double purpose, both to provide answers to "why" questions, and to identify the context in which an arbitrary help request is formed.

5 Conclusions and future work

We have presented a design of a help interface, that smoothly integrates user navigation and manipulation with system explanation. This interface utilizes explicit representation and presentation both of the actual domain knowledge, but also of contextual information.

We emphasised the need of a user-oriented design methodology for the success of intelligent help. This was illustrated by the strategy employed in the project, where we formulate the design requirements from three sources of information: an *a priori* investigation of the primary help needs, identification of relevant interface requirements from literature, and finally a prototype-and-test approach to the formulation of detailed requirements for explanations and additional functionalities.

Our current work focus on acquiring detailed help dialogues from users with specific help needs. These are acquired both with and without the use of the prototype, to identify such issues that we may have overlooked in our first design. In the next phase, we will investigate the usefulness of the different types of user- and contextual information for explanation generation.

Acknowledgements

This work was carried out under the PUSH project sponsored by Ellemtel Utvecklings AB and the SICS frame program.

References

[Bretan and Karlgren 93] BRETAN I, KARLGREN J., "The Role of Natural Language in Multimodal Interaction", in *Proceedings of the 1993 ERCIM Workshop on Multimedia HCI*, Nancy, France.

[Cohen 92] COHEN P.R., "The Role of Natural Language in a Multimodal Interface", in *Proceedings of Fifth Annual Symposium on User Interface Software and Technology*, Monterey, CA, 1992, pp. 143–149.

[Feiner and McKeown 91] FEINER S.K., MCKEOWN K.R, "COMET: Generating Coordinated Multimedia Explanations (video)", in *Proceedings of ACM CHI'91 Conference on Human Factors in Computing Systems*, 1991, p. 449-450.

[IMI 93] *Intelligent Multimedia Interfaces*, M. Maybury (ed), AAAI/MIT Press 1993.

[Lemaire and Safar 91] LEMAIRE B., SAFAR B., "Some necessary features for explanation planning architectures: study and

proposal", in *Proceedings of the AAAI Workshop on Comparative Analysis of Explanation Planning Architectures*, Anaheim, USA, July 1991, p. 15–26.

[Lemaire and Moore 94] LEMAIRE B., MOORE J., "An Improved Interface for Tutorial Dialogues: Browsing a Visual Dialogue History", in *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'94)*, Boston, April 1994.

[Maybury 91] MAYBURY M.T., "Planning multimedia explanations using communicative acts", in *Proceedings of the 9th AAAI Conference*, Anaheim, USA, July 1991, p. 61–66.

[Moore and Swartout 90] MOORE, J.D., SWARTOUT W.R., "Pointing: a way toward explanation dialogue", in *Proceedings of AAAI'90*, pp. 457–464.

[Wahlster et al. 91] WAHLSTER W., "Designing illustrated texts: how language production is influenced by graphics generation", in *Proceedings of the Fifth Conference of the European Chapter of the Association for Computationel Linguistics*, Berlin, avril 1991, p. 8–14.